In re Application of WANG et al.
Serial No. 09/447,501

AMENDMENTS TO CLAIMS

Please amend the claims as follows (*wherein additions are shown by underlining and deletions are shown by strikethrough in amended claims*):

1. (Currently amended): In a computer system, a method, comprising:

receiving a request from a kernel mode driver;

determining that the kernel mode driver is to be monitored;

re-vectoring the request to a kernel mode driver verifier; and

taking action in the kernel mode driver verifier to actively test the kernel mode driver for errors, wherein the kernel mode driver verifier is capable of testing the kernel mode driver by simulating a low resource condition including failing requests for memory pool allocation.

2. (Original): The method of claim 1 wherein receiving a request from a driver includes receiving a function call in a kernel component of an operating system.

3. (Original): The method of claim 1 wherein determining that the driver is to be monitored includes checking a registry setting.

4. (Currently amended): The method of claim 1 wherein the request from the driver includes a memory allocation request, and wherein taking action in the kernel mode driver verifier to test the driver includes allocating memory space thereto from a special pool of memory.

PAGE 6/25 * RCVD AT 11/29/2003 2:15:16 PM [Eastern Standard Time] * SVR:USPTO-EFXRF-2/0 * DNIS:7467238 * CSID:425-836-8957 * DURATION (mm-ss):07-34

5. (Currently amended): The method of claim 1 wherein the request from the driver includes a memory allocation request, and wherein taking action in the kernel mode driver verifier to test the driver includes marking memory bounding the memory space to detect improper access of the memory bounding the memory space.

6. (Currently amended): The method of claim 1 wherein the request from the driver includes a memory deallocation request, and wherein taking action in the kernel mode driver verifier to test the driver includes marking deallocated memory space to detect improper access of the deallocated memory space.

7. (Currently amended): The method of claim 1 wherein taking action in the kernel mode driver verifier to test the driver includes maintaining allocation information in at least one data structure associated with the driver.

8. (Original): The method of claim 7 wherein the request from the driver includes a memory allocation request, and wherein maintaining allocation information includes adding data corresponding to the allocation request to the data structure.

9. (Original): The method of claim 7 wherein the request from the driver includes a memory deallocation request, and wherein maintaining allocation information includes removing data corresponding to the allocation request from the data structure.

10. (Original): The method of claim 7 further comprising providing the allocation information to a user interface.

11. (Currently amended): The method of claim 1 wherein taking action in the kernel mode driver verifier to test the driver includes validating call parameters.

12. (Currently amended): The method of claim 1 wherein taking action in the kernel mode driver verifier to test the driver includes checking for resources allocated to the driver at driver unload.

13. (Currently amended): The method of claim 1 wherein taking action in the kernel mode driver verifier to test the driver includes simulating a low resource condition.

14. (Original): The method of claim 13 wherein simulating the low resource condition includes failing requests for memory pool allocation.

15. (Original): The method of claim 13 wherein simulating the low resource condition includes invalidating driver code and data.

16. (Currently amended): The method of claim 1 wherein taking action in the kernel mode driver verifier to test the driver includes checking for timers in deallocated pooled memory.

17–26. (Canceled)

27. (Currently amended): A computer-readable medium having computer-executable instructions, comprising:

receiving a request from a component for allocation of memory space;

determining a location of memory space to allocate;

restricting access to areas bounding the location, wherein any access request to at least one of the areas results in an access violation;

allocating the memory space; and

monitoring the areas bounding the location for an access violation.

28. (Original): The computer-readable medium of claim 27 having further computer-executable instructions, comprising, detecting an access violation.

29. (Currently amended): The computer-readable medium of claim 27 having further computer-executable instructions, comprising

receiving a request from the component for deallocation of the memory space;

restricting access to deallocated memory space, wherein any access request to the deallocated memory space results in an access violation; and

monitoring the deallocated memory space for an access violation.

30. (Original): The computer-readable medium of claim 29 having further computer-executable instructions, comprising, detecting an access violation in the deallocated memory space.

31. (Currently amended): A computer-readable medium having computer-executable instructions, comprising:

receiving a plurality of requests from a ~~component~~ driver for allocation of various distinct sets of memory;

allocating the memory;

tracking the ~~space~~ memory allocated to the ~~component~~ driver on each request;

receiving requests for deallocation of at least one of the sets of memory allocated to the ~~component~~ driver;

tracking the ~~space~~ memory deallocated in each received deallocation request; ~~and~~

determining from the tracking whether ~~space~~ memory remains allocated to the driver at a time when the driver should have no ~~space~~ memory allocated thereto;

and generating an error at the time if memory remains allocated.

32. (Currently amended): A computer-readable medium having computer-executable instructions, comprising:

receiving information corresponding to a driver unload;

determining whether resources remain associated with the driver;

and if ~~so~~ resources remain associated with the driver, generating an error.

33. (Original): The computer-readable medium of claim 32 wherein determining whether resources remain associated with the driver includes examining lists maintained by a system kernel.

34. (Original): The computer-readable medium of claim 32 wherein determining whether resources remain associated with the driver includes maintaining information tracking memory allocated to the driver and deallocated thereby.

35. (Currently amended): A system for monitoring drivers, comprising:

an operating system component including an interface for receiving requests from drivers;

a re-vectoring component for examining the requests to determine whether requesting drivers are to be monitored; and

a driver verifier component operably connected to the re-vectoring component, the driver verifier receiving information from the re-vectoring component for a driver that is to be monitored and executing at least one test to monitor the driver, wherein in response to a request for memory from the driver, the driver verifier component allocates memory for the driver to use from a pool of memory other than a memory pool normally allocated from when the driver is operating unmonitored.

36. (Original): The system of claim 35 wherein the operating system component comprises a kernel component.

37. (Original): The system of claim 35 wherein the re-vectoring component determines that the driver is to be monitored based on a setting in a registry.

38. (Original): The system of claim 37 further comprising a user interface for writing driver information to the registry.

39. (Currently amended): The system of claim 35 wherein ~~the~~ a request from the driver includes a memory allocation request, and wherein a test by the driver verifier includes allocating memory space thereto from a special pool of memory, and marking memory bounding the memory space to detect improper access of the memory bounding the memory space.

40. (Currently amended): The ~~method~~ system of claim 35 wherein ~~the~~ a request from the driver includes a memory deallocation request, and wherein a test by the driver verifier includes deallocating the memory space, and marking the deallocated memory space to detect improper access thereof.

41. (Original): The system of claim 35 wherein a test by the driver verifier includes examining resources allocated to the driver.

42. (Original): The system of claim 41 wherein examining resources allocated to the driver includes tracking outstanding memory allocated to the driver.

43. (Original): The system of claim 41 wherein examining resources allocated to the driver includes reviewing lists maintained by the operating system component for information therein associated with the driver.

44. (Currently amended): The ~~method~~ system of claim 35 wherein a test performed by the driver includes validating call parameters.

45. (Currently amended): The ~~method~~ system of claim 35 wherein a test performed by the driver includes failing requests for memory pool allocation.

46. (Currently amended): The ~~method~~ system of claim 35 wherein a test performed by the driver includes invalidating driver code and data.

47. (Currently amended): The ~~method~~ system of claim 35 wherein a test performed by the driver includes checking for timers in deallocated pooled memory.

48. (Currently amended): In a computer system, a method for verifying system components, comprising:

selecting one or more tests for verifying functionality of ~~the~~ a system component;

modifying a request for system services to include execution of the selected tests, wherein one of the tests includes restricting access to a resource, such that an attempted access to the resource causes an access violation;

executing the modified request; and

generating errors for any test failures.

49. (Currently amended): The method of claim 48 wherein the system component comprises a device driver.

50. (Previously presented): The method of claim 48 wherein the request for system services comprises a request to a kernel component.

51. (Previously presented): The method of claim 48 further comprising applying a test condition designed to detect a specific error.

52. (Previously presented): The method of claim 51 wherein applying the test condition includes restricting available system resources.

53. (Previously presented): A computer-readable medium having computer-executable instructions for performing the method of claim 48.